Trabalho 3 – Web Services (WS) ou Application Interface Programming (API)

Sistema de Locadora de Aparelhos de Festas

Equipe: André Alves de Freitas e Arthur Roberto da Silva

1. Objetivo

O objetivo deste projeto foi desenvolver um sistema de locadora de aparelhos com um servidor back-end em Java Spring Boot e dois clientes para interação com o sistema, sendo um cliente Python e um cliente Node.js. O sistema permite o gerenciamento de clientes e aparelhos, registro de locações e listagem de locações realizadas.

A solução foi projetada para ser modular e fácil de utilizar, permitindo a gestão de equipamentos para locação de forma eficiente, com funcionalidades como:

- **Cadastro de clientes**
- **Cadastro e visualização de aparelhos disponíveis**
- **Registro de locações**
- **Listagem de locações realizadas**

2. Estrutura do Sistema

O sistema foi dividido em duas partes principais: o **servidor** e os **clientes**.

2.1 Servidor: Locadora API (Java Spring Boot)

O servidor foi desenvolvido utilizando Java Spring Boot. Ele expõe uma API RESTful que gerencia todas as interações com o banco de dados(dados definidos no próprio servidor) e processa as requisições dos clientes. A API permite as seguintes funcionalidades:

1. **Gerenciar Clientes**
    - Cadastrar novos clientes.
    - Listar todos os clientes cadastrados.
2. **Gerenciar Aparelhos**
    - Cadastrar novos aparelhos.
    - Listar todos os aparelhos disponíveis para locação.
3. **Registrar Locação**
    - Registrar uma locação de um aparelho para um cliente.
    - A locação é registrada com as informações do cliente e do aparelho.

4. **Listar Locações**
    - Exibir todas as locações realizadas, mostrando o nome do cliente e os aparelhos alugados.

A comunicação com a API é feita via requisições HTTP GET e POST, que são manipuladas pelos métodos do LocadoraService no servidor.

2.2 Cliente Python

O cliente Python utiliza a biblioteca `requests` para enviar requisições à API do servidor. Ele permite a interação com as funcionalidades da locadora por meio de um menu interativo baseado em linha de comando. As opções do cliente Python incluem:

- **Gerenciar Clientes**: Listar clientes e adicionar novos.
- **Gerenciar Aparelhos**: Listar aparelhos e adicionar novos.
- **Registrar Locação**: Registrar uma locação de aparelhos para um cliente.
- **Listar Locações**: Exibir todas as locações registradas.

O cliente Python interage com a API para obter dados de clientes e aparelhos, e envia informações de locação para o servidor quando necessário.

2.3 Cliente Node.js

O cliente Node.js foi desenvolvido utilizando a biblioteca `axios` para enviar requisições HTTP para a API do servidor. Ele também oferece um menu interativo em linha de comando, utilizando o pacote `readline-sync` para capturar as entradas do usuário. O fluxo de funcionalidades é o mesmo do cliente Python, com a possibilidade de:

- **Gerenciar Clientes**: Listar e adicionar novos clientes.
- **Listar Aparelhos**: Exibir aparelhos disponíveis para locação.
- **Registrar Locação**: Registrar locações de aparelhos para clientes.
- **Listar Locações**: Exibir as locações realizadas.

O cliente Node.js tem a mesma lógica de interação que o cliente Python, mas foi desenvolvido em **JavaScript** e pode ser executado em ambientes que suportam Node.js.

3. Fluxo de Execução do Sistema

3.1 Servidor - Spring Boot

1. **Início**: O servidor API é inicializado e começa a escutar requisições na URL base configurada (`http://localhost:8080/api`).
2. **Gerenciamento de Clientes e Aparelhos**: O servidor permite o cadastro e a listagem de clientes e aparelhos, recebendo requisições **POST** para adicionar novos dados e **GET** para retornar informações existentes.

3. **Registro de Locação**: Quando um cliente solicita registrar uma locação, o servidor valida a existência do cliente e do aparelho, verifica a disponibilidade do estoque e, em seguida, registra a locação.
4. **Listagem de Locações**: O servidor permite que os clientes solicitem a listagem de todas as locações registradas, fornecendo um histórico completo.

3.2 Cliente Python e Node.js

1. **Menu Principal**: Ambos os clientes (Python e Node.js) apresentam um menu interativo para o usuário, com opções de gerenciar clientes, aparelhos, registrar locações ou listar locações.
2. **Gerenciamento de Clientes**: O usuário pode adicionar novos clientes ou listar os clientes existentes. Para adicionar um cliente, as informações são enviadas para a API do servidor via **POST**.
3. **Gerenciamento de Aparelhos**: O cliente permite a visualização dos aparelhos disponíveis para locação e a possibilidade de adicionar novos aparelhos. As requisições de listagem e adição são feitas via **GET** e **POST**.
4. **Registrar Locação**: O usuário pode registrar uma locação de um aparelho para um cliente. O sistema valida a existência do cliente e a disponibilidade do aparelho antes de realizar o registro.
5. **Listar Locações**: O cliente pode visualizar todas as locações realizadas, com informações sobre os clientes e os aparelhos alugados.

4. Tecnologias Utilizadas

- **Servidor**:
    - **Java Spring Boot**: Framework para a criação da API RESTful.
    - **Maven**: Gerenciamento de dependências e build do servidor.
- **Clientes**:
    - **Python**:
        - **requests**: Biblioteca para fazer requisições HTTP.
        - **readline-sync**: Biblioteca para interação com o usuário em linha de comando.
    - **Node.js**:
        - **axios**: Biblioteca para fazer requisições HTTP.
        - **readline-sync**: Biblioteca para interação com o usuário em linha de comando.

5. Conclusões

O sistema de locadora de aparelhos foi desenvolvido com sucesso utilizando o framework Spring Boot para o servidor, e Python e Node.js para os clientes. As funcionalidades de gerenciamento de clientes, aparelhos e locações funcionaram conforme esperado, permitindo a interação com a API do servidor.

A estrutura do código é modular, permitindo fácil expansão e manutenção. O uso de **API RESTful** para comunicação entre o servidor e os clientes assegura que o sistema seja flexível e fácil de integrar com outros sistemas ou interfaces no futuro.